School of Computer Science
Dr. Ying Zhou

COMP5338: Advanced Data Models Sem. 2/2024

MongoDB Project

26.08.2024

1 Introduction

This assignment tests your ability to model MongoDB data and write complex aggregation workloads. The data set you will work with consists of climate data objects, city objects, and a co2 object.

2 Data set

The data set consists of three JSON files. The `climates.json` file is the same JSON file you encountered in Assignment 1. The `coords.json` file contains many JSON objects where each object stores latitude, longitude, and elevation data for a particular city. This information is extracted from relevant Wikipedia articles for these cities. An example of such an object is as follows:

```json
{
    "_id": 1,
    "city_id": 2,
    "city": "Athens",
    "latitude": 37.98,
    "longitude": 23.73,
    "elevation": 153
}
```

Each JSON object in `coords.json` contains the following fields:

- `_id`: A unique integer identifier;
- `city_id`: The unique integer identifier for the object where climate data for the city can be found;
- `city`: The name of the city;
- `latitude`: The city' s latitude in degrees;

- `longitude`: The city' s longitude in degrees; and
- `elevation`: The city's elevation in metres.

For the latitude fields, positive values indicate locations in the northern hemisphere while negative values indicate locations in the southern hemisphere. A latitude of zero corresponds to the equator.

For the longitude fields, positive values indicate locations in the eastern hemisphere while negative values indicate locations in the western hemisphere. A longitude of zero corresponds to the Greenwich meridian (the prime meridian); a longitude of 180 or -180 corresponds to the antimeridian (we will assume that the Earth is a perfect sphere for simplicity).

For example, the above `coords` JSON object gives the following location information on Athens:

- In the `climates` collection, the JSON object whose _id is 3 contains climatic information on this city;
- Athens's latitude is 37.98 °N;
- Athens's longitude is 23.73 °E; and
- Athens's elevation is 153 metres.

The `co2.json` file contains emissions-related data, population data, and gross domestic product data for each region represented in the climates data set. Data points are given for various years.

This file consists of one large document with 44 fields. Each field corresponds to a region. For each region, there is an embedded document that consists of only two fields "iso_code" and "data".

The "data" field contains an array of embedded documents. These embedded documents contain fields storing emissions, population, and/or gdp information for a particular (region, year) pair. The calendar year is indicated in the embedded field "year". Note that some embedded documents for certain (region, year) pairs may be missing some fields.

An example of an embedded document from the data array for the region *Switzerland* is as follows:

```
{
    "year": 2005,
    "population": 7428437,
    "gdp": 371135479808,
    "cement_co2": 1.8474559783935547,
    "cement_co2_per_capita": 0.24870049953460693,
    "co2": 45.77828598022461,
    "co2_growth_abs": 0.5468800067901611,
    "co2_growth_prct": 1.2090682983398438,
    "co2_including_luc": 45.63539123535156,
    "co2_including_luc_growth_abs": 0.4245759844779968,
    "co2_including_luc_growth_prct": 0.9391069412231445,
    "co2_including_luc_per_capita": 6.143337249755859,
    "co2_including_luc_per_gdp": 0.12296154350042343,
    "co2_including_luc_per_unit_energy": 0.1339138001203537,
    "co2_per_capita": 6.162573337554932,
    "co2_per_gdp": 0.12334656715393066,
    "co2_per_unit_energy": 0.13433311879634857,
    "coal_co2": 0.6332970261573792,
    "coal_co2_per_capita": 0.08525305986404419,
    "consumption_co2": 108.47098541259766,
    "consumption_co2_per_capita": 14.602127075195312,
    "consumption_co2_per_gdp": 0.29226788878440857,
    "cumulative_cement_co2": 109.25566864013672,
    "cumulative_co2": 2413.0439453125,
    "cumulative_co2_including_luc": 2668.85302734375,
    "cumulative_coal_co2": 582.9473876953125,
    "cumulative_flaring_co2": 0.6536980271339417,
    "cumulative_gas_co2": 122.80497741699219,
    "cumulative_luc_co2": 285.35302734375,
    "cumulative_oil_co2": 1593.5401611328125,
    "cumulative_other_co2": 3.8419458866119385,
    "energy_per_capita": 45875.3125,
    "energy_per_gdp": 0.9182141423225403,
    "flaring_co2": 0.05752100050449371,
    "flaring_co2_per_capita": 0.007743351627141237,
    "gas_co2": 6.611806869506836,
    "gas_co2_per_capita": 0.8900670409202576,
    "ghg_excluding_lucf_per_capita": 7.433596134185791,
    "ghg_per_capita": 7.185899257659912,
    "land_use_change_co2": -0.14289599657058716,
    "land_use_change_co2_per_capita": -0.019236348569393158,
    "methane": 5.550000190734863,
    "methane_per_capita": 0.7471289038658142,
    "nitrous_oxide": 2.5899999141693115,
    "nitrous_oxide_per_capita": 0.3486601412296295,
    "oil_co2": 36.371177673339844,
    "oil_co2_per_capita": 4.8962082862854,
    "other_co2_per_capita": 0.034600820392370224,
    "other_industry_co2": 0.25703001022338867,
    "primary_energy_consumption": 340.7818603515625,
    "share_global_cement_co2": 0.1916566789150238,
    "share_global_co2": 0.15469765663146973,
    "share_global_co2_including_luc": 0.13233783841133118,
    "share_global_coal_co2": 0.005481316242367029,
    "share_global_cumulative_cement_co2": 0.48196104168891907,
    "share_global_cumulative_co2": 0.20378369092941284,
    "share_global_cumulative_co2_including_luc": 0.1405131071805954,
    "share_global_cumulative_coal_co2": 0.10101854056119919,
    "share_global_cumulative_flaring_co2": 0.005036015529185533 5,
    "share_global_cumulative_gas_co2": 0.08332201838493347,
    "share_global_cumulative_luc_co2": 0.039642538875341415,
    "share_global_cumulative_oil_co2": 0.38045552372932434,
    "share_global_cumulative_other_co2": 0.07439224421977997,
    "share_global_flaring_co2": 0.016529820859432 22,
    "share_global_gas_co2": 0.12301123142242432,
    "share_global_luc_co2": -0.0029210711363703012,
    "share_global_oil_co2": 0.32739678025245667,
    "share_global_other_co2": 0.10609104484319687,
    "share_of_temperature_change_from_ghg": 0.12081058323383331,
    "temperature_change_from_ch4": 0.00027447790489532053,
    "temperature_change_from_co2": 0.0011337921023368835,
    "temperature_change_from_ghg": 0.0015000600833445787,
    "temperature_change_from_n2o": 0.00009178999607684091,
    "total_ghg": 53.380001068115234,
    "total_ghg_excluding_lucf": 55.220001220703125,
    "trade_co2": 62.69269561767578,
    "trade_co2_share": 136.9485321044922
}
```

You can find an explanation of the meaning of the above field names at https://github.com/owid/co2-data/blob/master/owid-co2-codebook.csv. Note that the JSON file you are given is adapted from data that is freely available at https://github.com/owid/co2-data/tree/master.

3 Data Modelling Requirements

You are asked to store all information in the three JSON files in a MongoDB database called *emissions*. You need to decide on the collections you will use to store the data and what a typical document in each collection would look like. A reasonable model would have three to four collections. The decision should be based on the feature of the data and the typical workloads you will run against the data. This assignment focuses on analytic workloads; you should also consider transactional workloads involving basic CRUD (Create, Retrieve, Update and Delete) operations when designing your collections.

The JSON files contain redundant data. For instance, a city's name appears in both `climates.json` and `coords.json` files. MongoDB does not require normalisation; a certain level of redundancy can be beneficial. However, you still need to make decisions on whether or not you want to keep or remove certain redundancies.

The `climates.json` and `co2.json` files contain incomplete data. You will need to decide how to model missing values. In some cases, you may omit the field(s) in the document with missing values. In other cases, you may find it more convenient to keep the field(s) but assign a value of `null`, an empty string, or an empty array.

Your document should not contain any additional information beyond those in the JSON files. In particular, you should not store pre-computed values that will be used in the aggregation workloads.

You can use a Python script and/or one MongoDB aggregate command to pre-process the data before importing them into MongoDB collections.

4 Aggregation Workloads

W1 Find the number of regions in the `co2` data set that have a per capita gdp of 20000 international dollars or more in the year 2022. The per capita gdp for a region can be obtained by dividing the region's gdp by its population.

W2 Find all cities in the northern hemisphere whose average monthly rainfall for June is greater than their average monthly rainfall for March. Your result should be an array consisting of documents in the following format:

```
{
  city: city_name,
  latitude: latitude
}
```

where `city_name` is the name of the city; and

`latitude` is the city's latitude.

Your results must be sorted by latitude in ascending order.

W3 Find all cities with a latitude above 30 degrees North or South that are located in the top five regions that have experienced the greatest absolute per capita growth in overall co2 emissions during the period 1970 to 2020. For each relevant city, the output must be in the following format:

```
{city_name: <name of a city>,
 co2_growth: <absolute per capita co2 increase between
              1970 and 2020>}
```

W4 Create a new collection called *analytics1* where each document represents a (region, year) pair. Each document must contain only the following fields:

- _id: A unique value (you can define it however you wish)
- id: The name of a region
- year: The year
- co2: As given in the original data set for the relevant (region, year) pair
- coal_co2_cont: The percentage of co2 emissions due to coal
- gas_co2_cont: The percentage of co2 emissions due to gas
- oil_co2_cont: The percentage of co2 emissions due to oil
- other_co2_cont: The percentage of co2 emissions due to other causes

W5 Add a field main_co2_cont to each document that is concerned with the year 2022 in the *analytics1* collection. This field must be set to one of the following strings indicating the main contributor of co2 emissions for the relevant region in the year 2022:

- "coal"
- "gas"
- "oil"
- "other"

W6 Considering only the regions that have at least one city with an elevation greater than 500 metres as listed in the *coords* data set, find the three regions that have the lowest average annual co2 emissions during the years 2001 to 2022 inclusive. Your result must consist of three documents in the following format:

```
{region: <name of a region>,
 co2_emissions: <a 22-element array of numbers
                representing annual co2 emissions
                for each year>,
 average_emissions: <average annual co2 emissions
                    during the years 2001 to 2022
                    inclusive>}
```

W7 Create a new collection called *analytics2* where each document represents a region. Each document must contain only the following fields:

- _id: The name of a region
- estimated_annual_energy_y01_y21: An array consisting of values representing estimated annual energy consumption for each year in the period 2001 to 2021 inclusive

The estimated annual energy consumption in a given year X is given by the following formula: energy_per_capita X population

W8 Create a new collection called *analytics3* where each document represents a region for the year 2022. Each document must contain only the following fields:

- _id: The name of a region
- co2: As given in the original data set for the relevant (region, year) pair
- data: An array consisting only of fields from the original co2 data set that contain "per_capita" in their names (do not hard code these names in your solution)

5 Implementation Requirements

You need to import the data into a database called "emissions". **Your documents and collections should be designed following the data modelling requirements.**

If pre-processing is involved, it should be implemented with a MongoDB aggregate statement and/or one Python script using only commonly available libraries. The pre-processing should only change the format of the input data set.

You are required to provide data import command(s) to prepare the collections according to your own schema. You are also required to set up proper indexes that will improve the performance of your queries. The import command(s) and the index creation command(s) should be written in a single `preparation.txt` file.

Implement each aggregation using a single MongoDB `aggregate` command. You **must not** use other commands such as `find`. Each workload implementation must be placed in a separate JavaScript file using the following filenames:

- w1.js - W1 implementation;
- w2.js - W2 implementation;
- w3.js - W3 implementation;
- w4.js - W4 implementation;
- w5.js - W5 implementation;
- w6.js - W6 implementation;
- w7.js - W7 implementation; and
- w8.js - W8 implementation

No other files are permitted (other than the driver file `a2_driver.js`). You must not modify the driver file.

Your JavaScript files for workloads must be able to execute correctly when they are called from the driver file.

To test your solutions, you should run the driver file using the following command:
`mongosh a2_driver.js`

It will print out the results of each query.

For workloads W1, W2, W3, and W6, you must store your result in a variable called `res`. You need to use `var res = db.collection.aggregate()` to ensure the variable `res` is updated properly in the relevant driver script.

Do not include configuration statements in your JavaScript files; include only the aggregation statements in your JavaScript scripts. The driver files contain all the required configuration statements.

6 Report

You are asked to prepare a report to describe your schema. You are also asked to create some indexes that would improve the performance of your queries and explain the query plans of all of your queries with such indexes.

The report should have the following sections:

- Introduction
- Schema Design
- Index and Query Execution
- Conclusion

The introduction and conclusion sections should be very brief. No marks are explicitly assigned for these sections. However, they contribute to the professionalism of the entire report, which will be assessed. Details of the two middle sections are given below.

6.1 Schema Design

In this section, briefly describe your collections and documents and how various relationships are implemented. For each collection, you must show a sample document with a full set of the fields.

6.2 Index and Query Execution

In this section, describe the indexes you have created and explain their usage in each query. You may include screenshots of MongoDB's execution plan output.

7 Deliverables and Submission Rules

There are two deliverables for this assignment. They should be submitted to the appropriate submission inboxes.

The deliverable for the first part is a single zip file that contains the txt file with your data import and index creation commands, the JavaScript files of your solutions as described in section 3 and, if applicable, a Python script and/or a text file containing one MongoDB aggregate command for pre-processing the data. **Do not include driver files, data files or input files** in your submission. Submit your zip file to the appropriate Assignment 2 submission inbox in Canvas.

The deliverable for the second part is a single pdf file containing your report. You should submit this pdf file to the appropriate submission inbox.

The submission deadline is week 7 13 September 2024, 23:59 Sydney time. Late penalties apply for late submissions.

Your script will be marked against a slightly different data set. The marking data set has the same format as the one given.

8 Resources

The following files are provided together with the assignment instruction:

- Data Files: `climates.json`, `coords.json`, and `co2.json`.
- Driver: `a2_driver.js`
- Dummy implementation for all workloads: `dummy.zip`

9 Automated Writing and Generative AI Tools Usage Guidelines

You are permitted to use automated writing and generative AI tools in the following cases:

- Spell checking and grammar checking of text that you wrote yourself
- Obtaining recommended readability improvements to text that you wrote yourself
- Obtaining translations of individual words or short phrases (but not entire sentences)

Do not post confidential, private, personal, or otherwise sensitive information into these tools. If you use these tools, you must be aware of their limitations, biases, and propensity for fabrication. Your use of these tools must adhere to the Student Charter 2020, including upholding honesty, ethics, professionalism, and academic integrity. Ultimately, you are 100% responsible for your assessment submission.

All uses of automated writing and generative AI tools must be appropriately acknowledged. You can do this by including an acknowledgment section at the end of your report where you need to identify the tool(s) that you have used, explain the purpose of using this tool, state verbatim the prompt(s) you have provided, state verbatim the output, and explain how you have used or adapted this output.